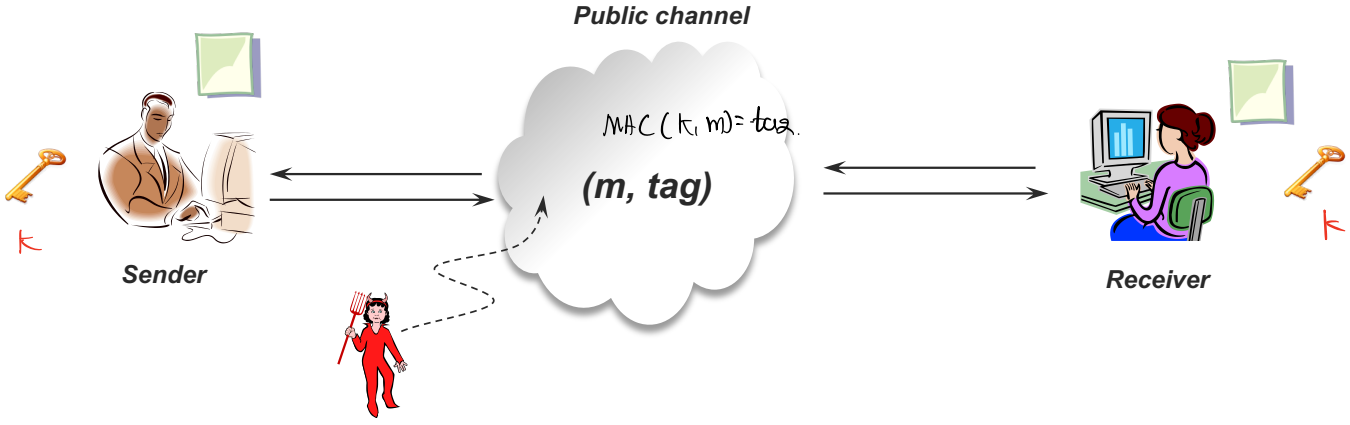
Digital Signature Schemes

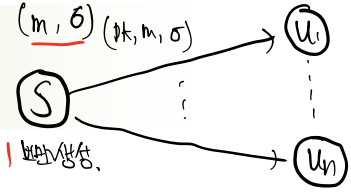
Jong Hwan Park

What can be done with MAC?

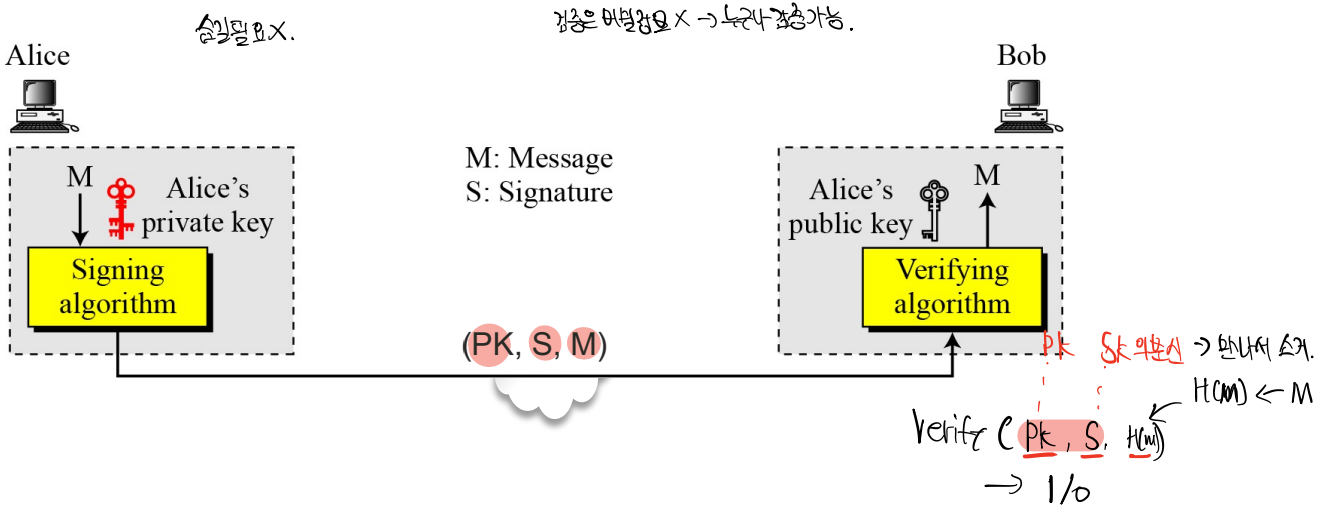


- What MAC achieved is to ensure that:
  - (a) The message is sent from the very sender
  - (b) The message is not changed during transmission
- In most situations, need more than two properties:
  - What if server releases a software patch to all of clients (using MAC) ?
  - How can a judge be convinced of the validity of (m, tag)?

Digital Signature Scheme (DSS)



- Definition of DSS = (KeyGen, Sign, Verify) 수학적으로 명확.
  - KeyGen(1^n) → a public key (PK) and a private key (SK)
  - Sign(SK, M) → a signature (S) on a message (M)
  - Verify(PK, S, M) → output 1(valid) or 0(invalid)



  - PK is the signer(Alice)'s public key
    - How can receiver(Bob) obtain PK? pk를 함께 전달.
    - Need authenticated channel for distributing PK (why?) Alice 까지만 확인하면 됨.

What can be done with DSS?

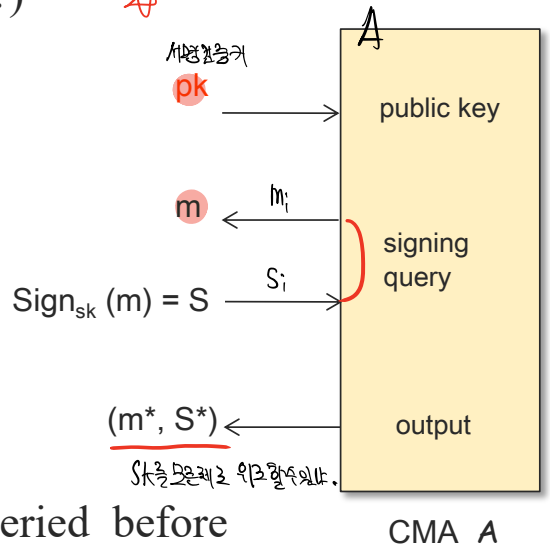
- As in MAC, DSS provides:
  - (a) sender authentication
  - (b) integrity

- More properties that DSS gives
  - (c) public verifiability (why?) 공개적으로 검증가능. / MAC은 대칭키를 가진이만
    - If signature S is valid, all other parties also verify it as valid
  - (d) transferability (← public verifiability)
    - Signature can be copied and shown to other parties
  - (e) non-repudiation (부인봉쇄) MAC과의 중요한 차이점
    - Signer cannot later deny having signed a message M
    - Meaning a receiver can prove a judge that a signer did certify M
    - Why is this not the case using MAC?
  - Not confidentiality!

(pk, S, M) → 다른이에게 전달가능.

Security of DSS

- Adversary is given a public-key PK, meaning that:
  - Adversary can verify (m, S) for free (why?)
  - Defining chosen-message attack (CMA):
  - What is the goal of adversary?
    - Forgery of signature

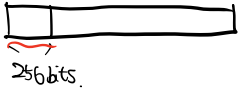


- Adversary breaks DSS iff
  - (a) Verify(m*, S*) =1 & (b) m* is not queried before

- *Existentially* unforgeable under chosen message attacks
  - vs. selective forgery
  - Why is the existential unforgeability meaningful? (random messages)

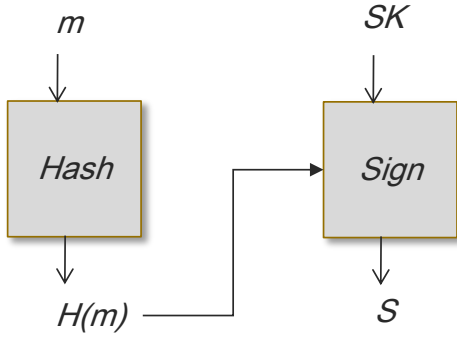
Hash-then-Sign Paradigm

- As in PKE, DSS is very inefficient when handling long messages
  - Idea: sign a hash value H(m) instead of a long message



- How it works:
  - Gen(1^n): outputs (PK, SK)
  - Sign(SK, m): obtain H(m) and S=Sign'(SK, H(m))
  - Vrfy(PK, m, S): obtain H(m) and
    Vrfy(PK, H(m), S) = 1 or 0

H가 충돌저항성을 가져야한다



- **Theorem:** If H is collision-resistant, the sig=(Gen, Sign, Vrfy) is existentially unforgeabile under chosen message attacks
  - What if H is not collision-resistant?

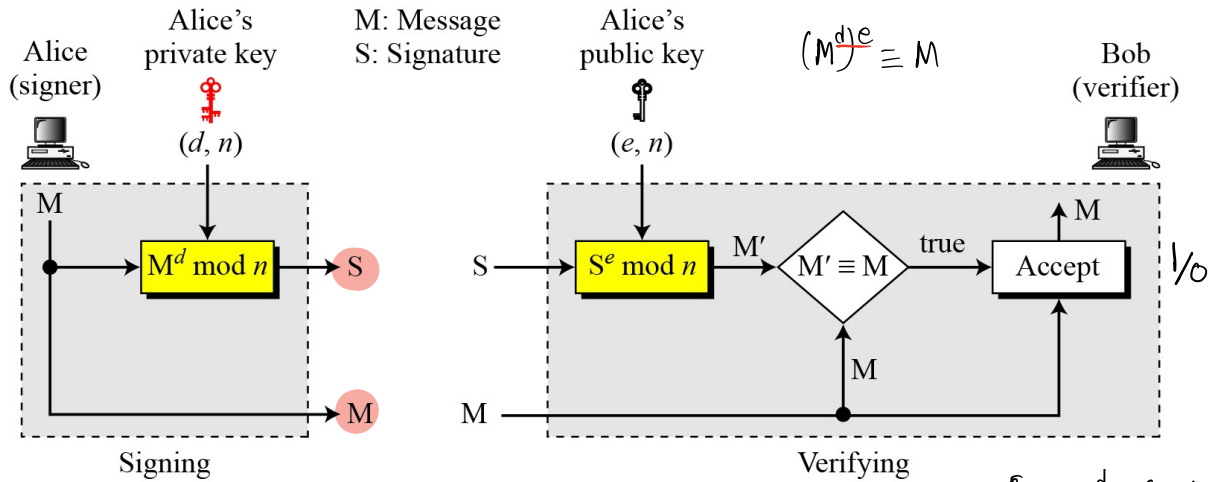
Sign(Sk, m_1) → S $H(m_1) = H(m_2)$

- General & standard way!

Verify(PK, m_2, S) → 1/0

Textbook RSA Signature

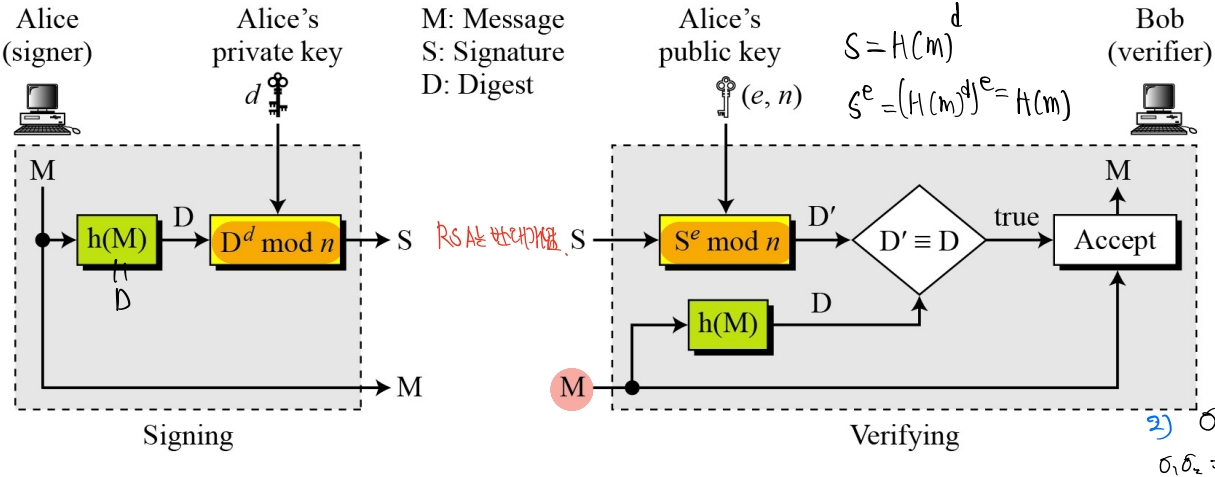
- Key generation same as in PKE
  - PK = (n, e) and SK = (d)
  - How it works:



- Is it secure?
  - 1) No-message attack (select S, $S^e \bmod n$)
  - 2) Known-message attack (given m_1^d, $m_2^d \bmod n$ → $(m_1 m_2)^d \bmod n$

Hashed RSA Signature

- How it works:

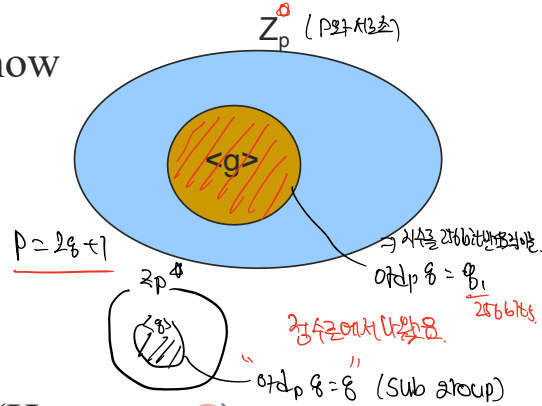


- How can we prevent the two previous attacks?
  - No-message attack / known-message attack
  - Theorem: hashed RSA signature is existentially unforgeable under CMA

- In PKCS#1: $H(M) = \left[00||01||FF||\cdots||FF||h(m)\right]^d \pmod n$

Digital Signature Algorithm (DSA)

- In 1991, DSA was proposed by NIST
  - Has been proved secure and widely used until now
  - Advantages:
    - Shorter size of signature (why ?)
    - Faster signature generation



- How it works:
  - Gen: PK = (H, p, q, g, $y=[g^x \bmod p]$) and SK= (H, p, q, g, x)
  - Sign: select a random $k \leftarrow Z_q^*$ and set

$$r := [\,[g^k \bmod p] \bmod q\,] \quad \text{and} \quad s := [\,(H(m)+xr)\cdot k^{-1} \bmod q\,]$$

signature = (r, s)

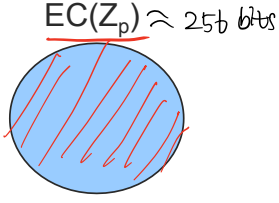
  - Vrfy: output 1 iff $r =? [\,[(g^{H(m)} y^r)^{1/s} \bmod p] \bmod q\,]$

- Differently chosen k should be used at each signing (why?)

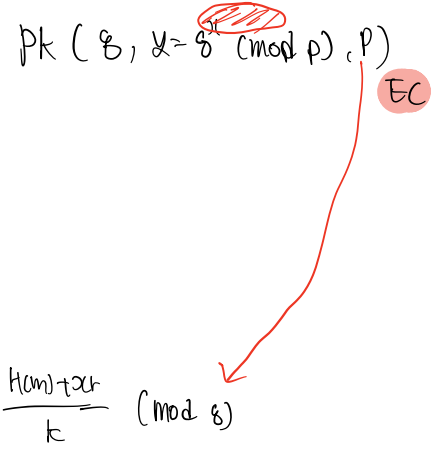
ECDSA

$p = 2q_1 \cdot q_2 + 1$

- DSA based on Elliptic Curve
  - Very efficient and widely used recently
  - Suitable for resourced-constraint IoT devices



- How ECDSA works:
  - ECDSA-Setup(1^n)
    - PK = (G, q, g, $y=g^x$), sk = (x)
  - ECDSA-Sign(sk, m)
    - Pick random $k \leftarrow Z_p$ and set $g^k \in G$
    - $r = F(g^k) \pmod q$ and $s = k^{-1}(H(m) + xr) \pmod q$
    - $\sigma = (r, s) \in Z_p \times Z_p$
  - ECDSA-Verify(σ, PK, m)
    - Compute $w = s^{-1} \pmod q$
    - $A = (g^{H(m)} y^r)^{1/s}$ and output 1 iff $r = F(A) \pmod q$



fixed base

$g^2, g^4 \ldots g^{2^t}$

g^k

Digital Signature Applications

- Public key certificates
  - Public key management & PKI
- Secure file transfer
  - Integrity check before execution
- User authentication
  - Server authentication (in SSL/TLS)
- Authenticated key exchange
  - DS+DH, DS+PKE: auth. & key exchange (in SSL/TLS, IPsec, SSH)
- Online banking/shopping
  - Non-repudiation on transactions
- Authorization in organizations
  - Signing m=(PK||user||access right) instead of m=(PK||user)
- And others …

Public Key Distribution

- PK is necessary for PKE, DSS, authentication, key agreement,…
- But, how can Alice convince Bob that she is the owner of PK_A?

- Alice and Bob cannot solve the PK distribution problem
  - Need a trusted third party (called a Certificate Authority (CA))
  - PK can then be distributed via an *authenticated* channel

Public Key Distribution (1)

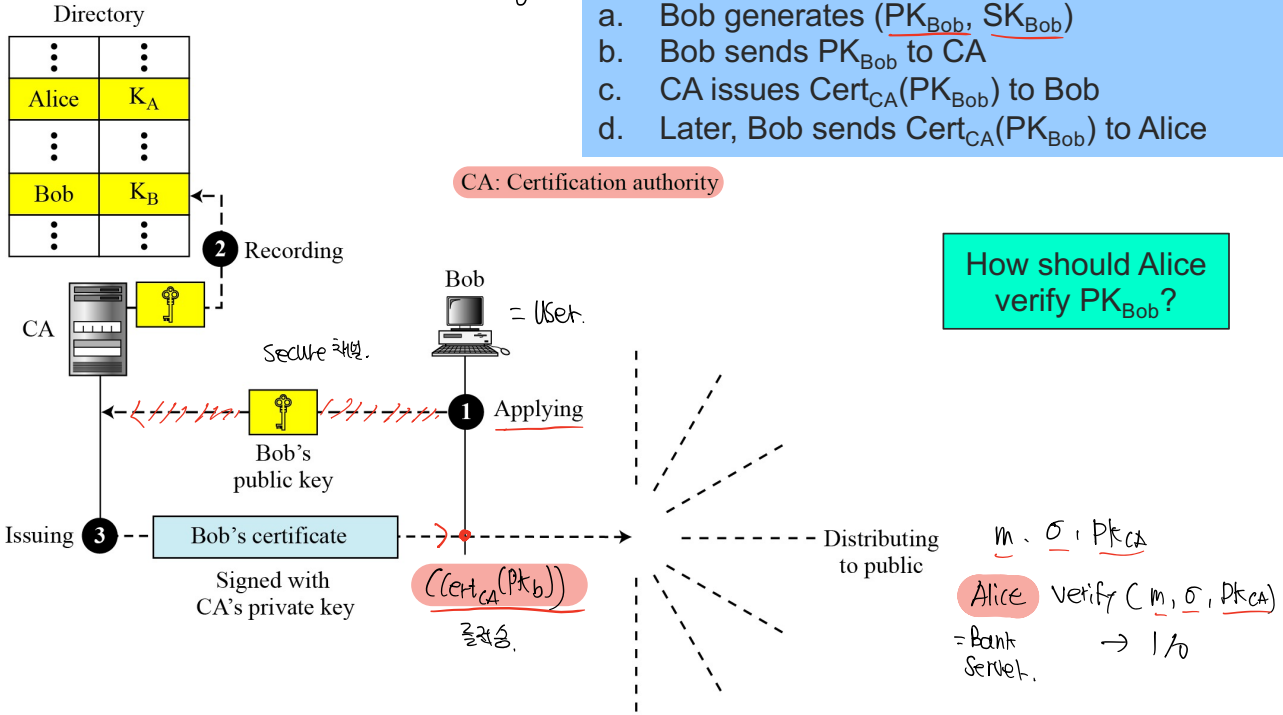
- Certificate Authority (CA)
  - CA generates its own keys (PK_{CA}, SK_{CA})
  - $Cert_{CA}(PK_{Bob}) = PK_{Bob} \| Bob \| Sign(SK_{CA}, PK_{Bob} \| Bob)$

Treat it as a message signed

a. Bob generates (PK_{Bob}, SK_{Bob})
b. Bob sends PK_{Bob} to CA
c. CA issues $Cert_{CA}(PK_{Bob})$ to Bob
d. Later, Bob sends $Cert_{CA}(PK_{Bob})$ to Alice

CA: Certification authority

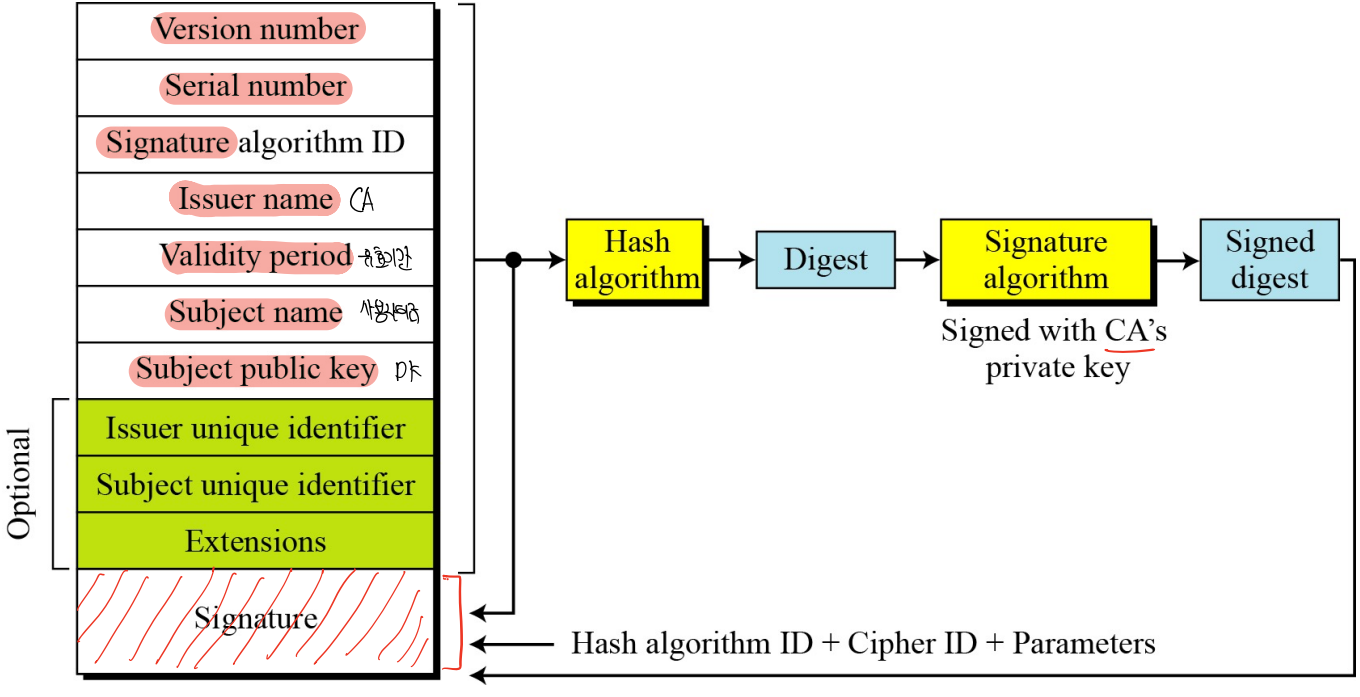
How should Alice verify PK_{Bob}?



Public Key Distribution (2)

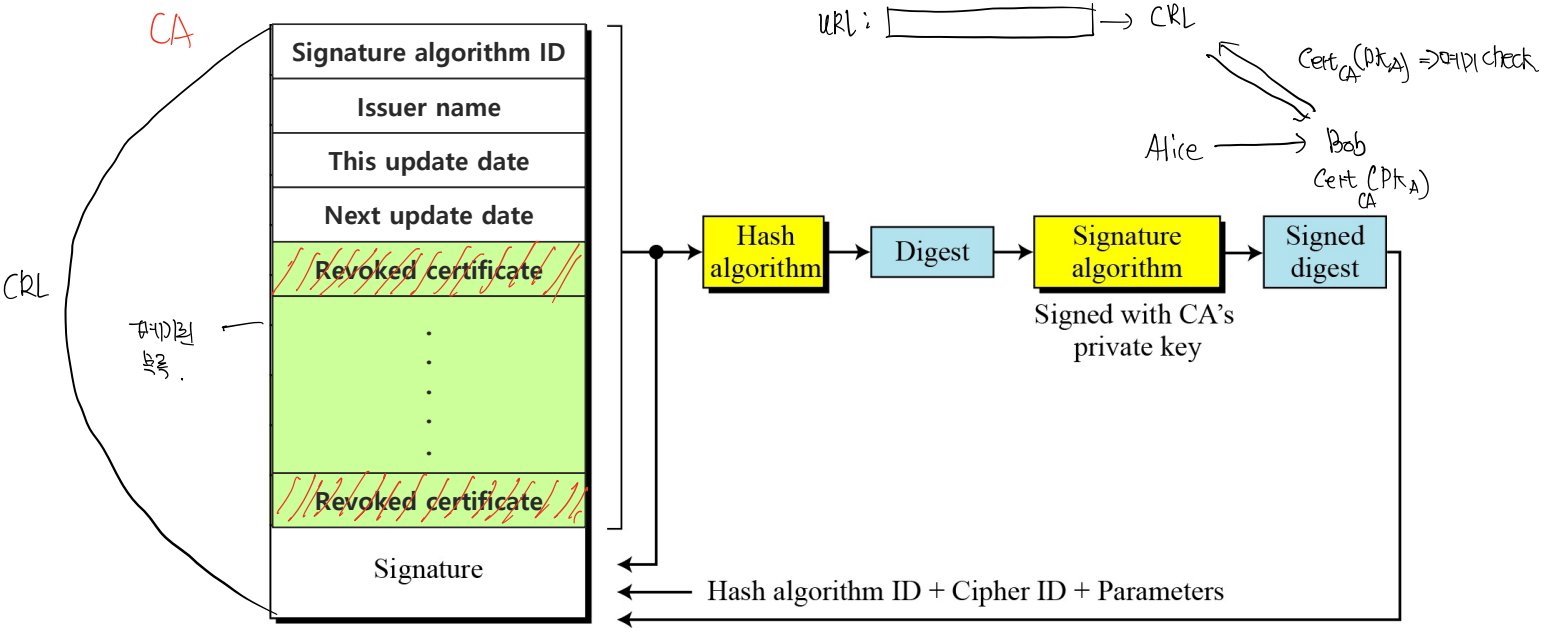
- X.509
  - Each certificate may have a different format
  - Designed by ITU, as a way to describe certificate in a structured way

$Cert_{CA}(PK_B) = PK_B \| Bob \| sign(SK_{CA}, m)$

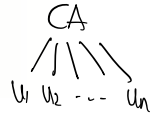


Public Key Distribution (3)

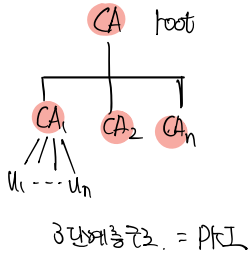
- Certificate Revocation List (CRL)
  - In some cases, a certificate must be revoked before expiration
    - Compromising user's SK, losing USB, formatting hard disk
    - Being laid off in an organization,…
  - Need to check whether or not a certificate is revoked before using it

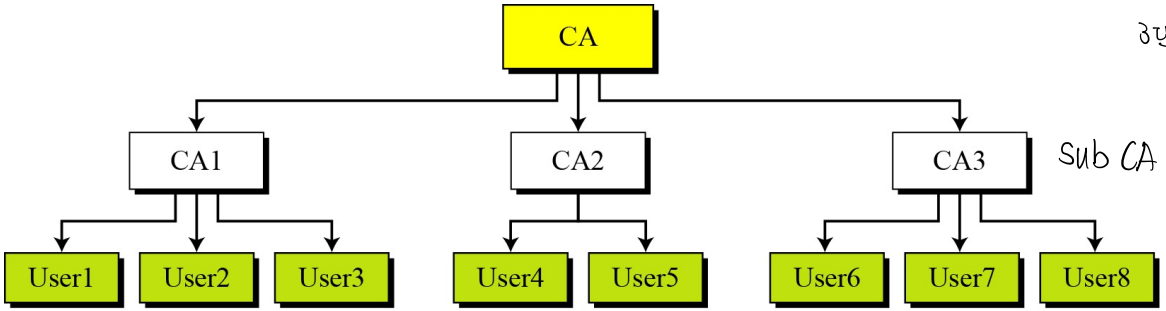


Public Key Infrastructure (PKI)



- Model for creating, distributing, and revoking certificates based on X.509
  - Single CA is unrealistic
  - In practice, there are multiple CAs in hierarchical model





X ⟶ Y
means X has signed a certificate for Y

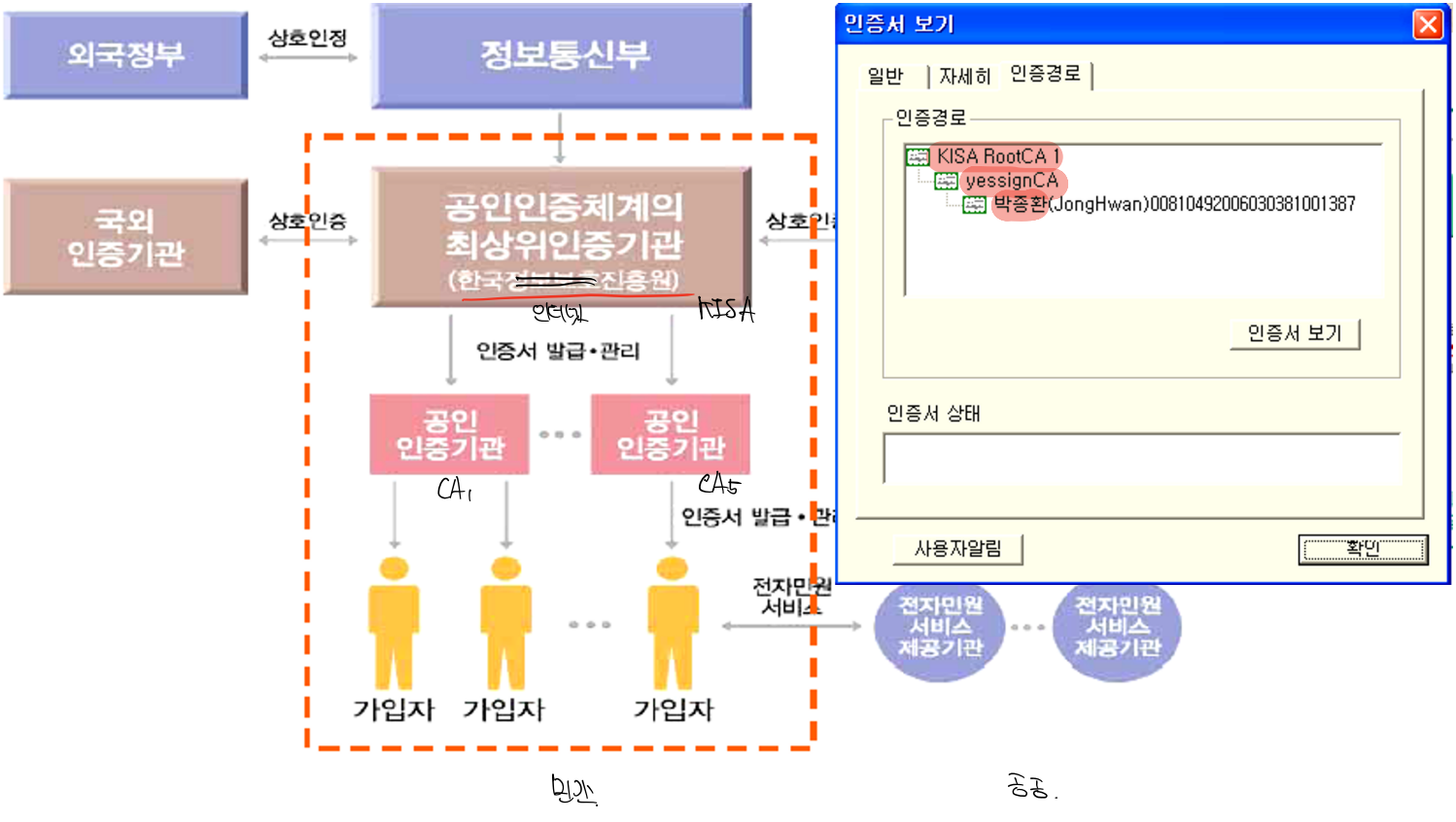
  - Delegation and Certificate chains
    - $Cert_{CA1}(PK_{u1})$, $Cert_{CA}(PK_{CA1})$, $Cert_{CA}(PK_{CA})$

Who certifies?
→ Self-signed Cert

⇒ 3개의 cert가 필요.
Self-Signed Cert.
PK_{CA1}
PK_{CA}

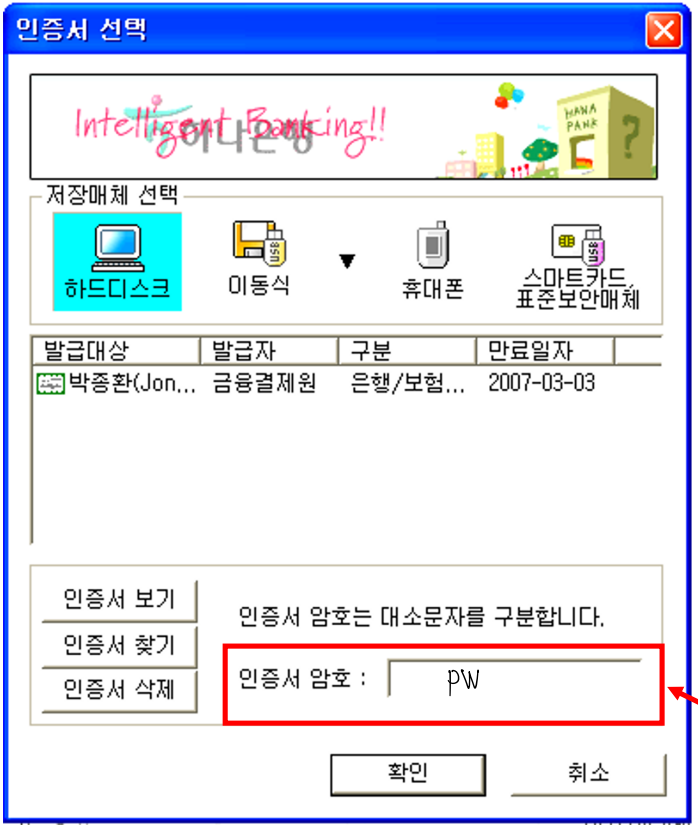
PKI in Korea

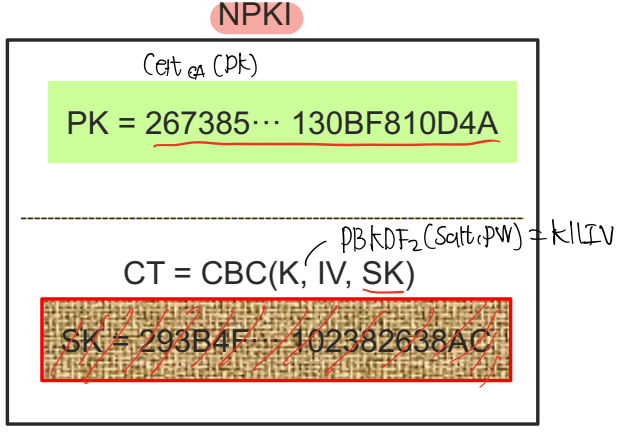
- Example of Certificate chains



How SK is stored (Korean Version)

- How to store a private key (SK) in NPKI



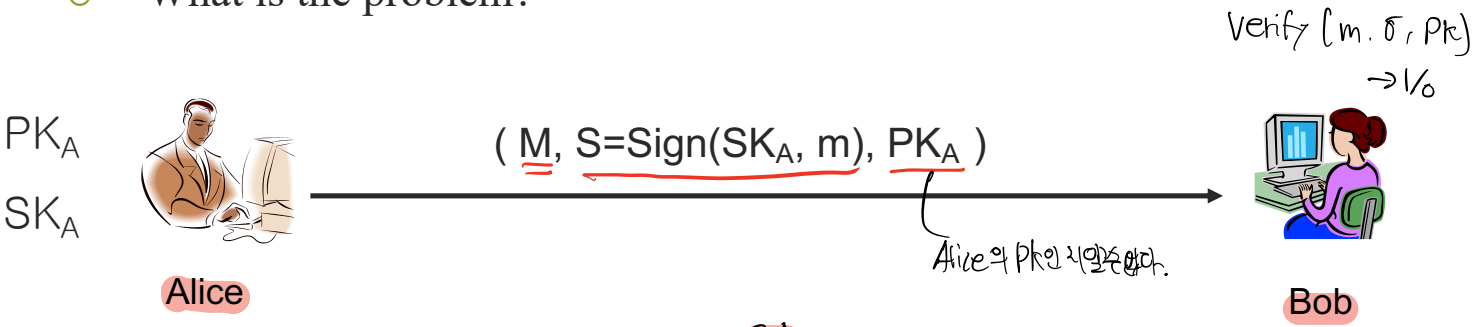


a. (K, IV) = KDF(PW, salt, iteration number)
b. SK = Decrypt (CT) using (K, IV)
c. Sign a message using SK

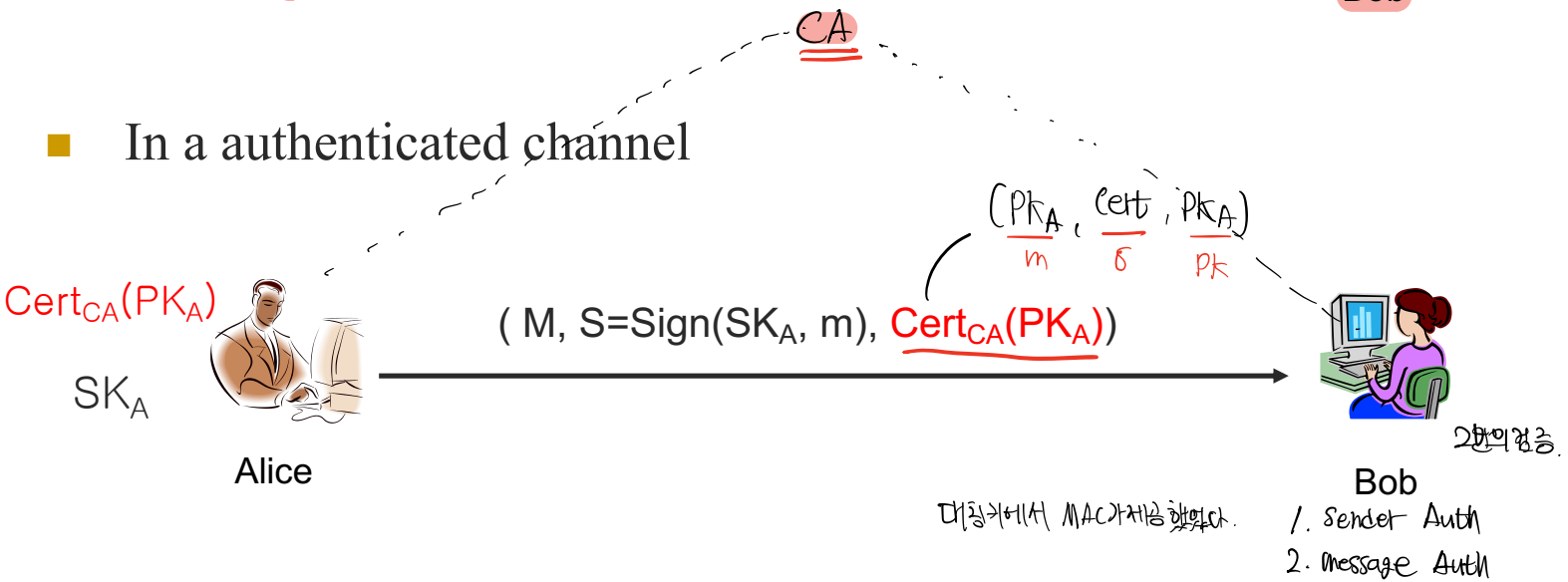
Why is that?

File Transfer via Authenticated Channel

- Alice wants to send a file (or message) to Bob
  - What is the problem?

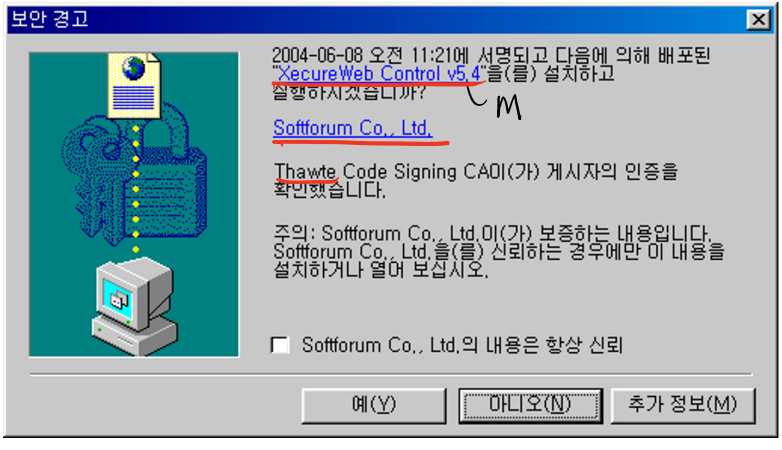


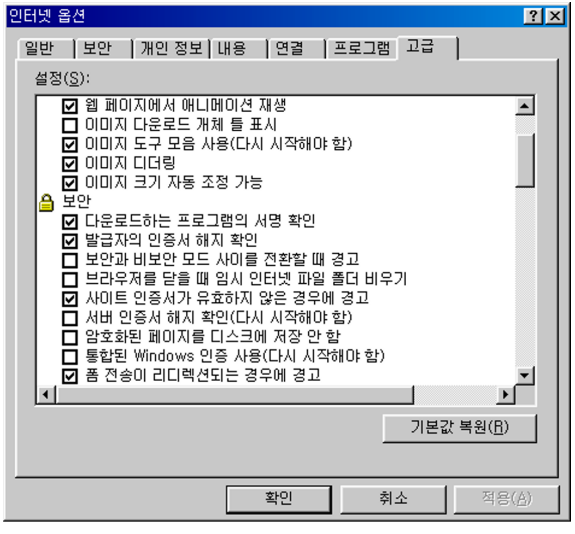
- In a authenticated channel

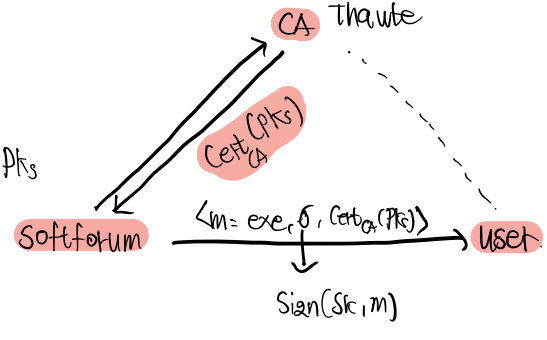


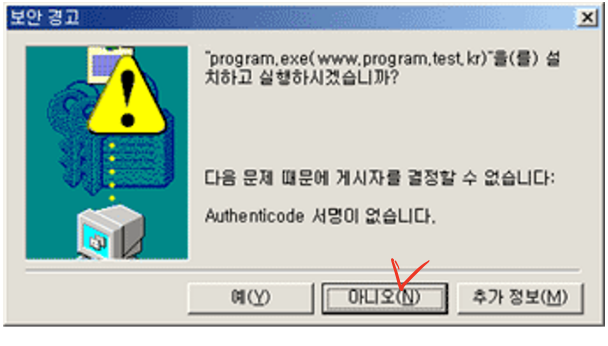
Integrity Check using DSS (1)

- Software distribution





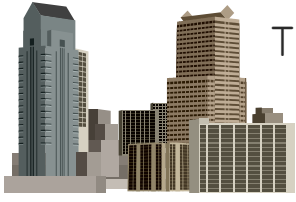




Integrity Check using DSS (2)

- What can we know from the notice?

Cert(for code-signing)



Thawte CA

"If signature is valid under the Cert (issued by Thawte CA), do you install the file sent by Softforum Co.?"

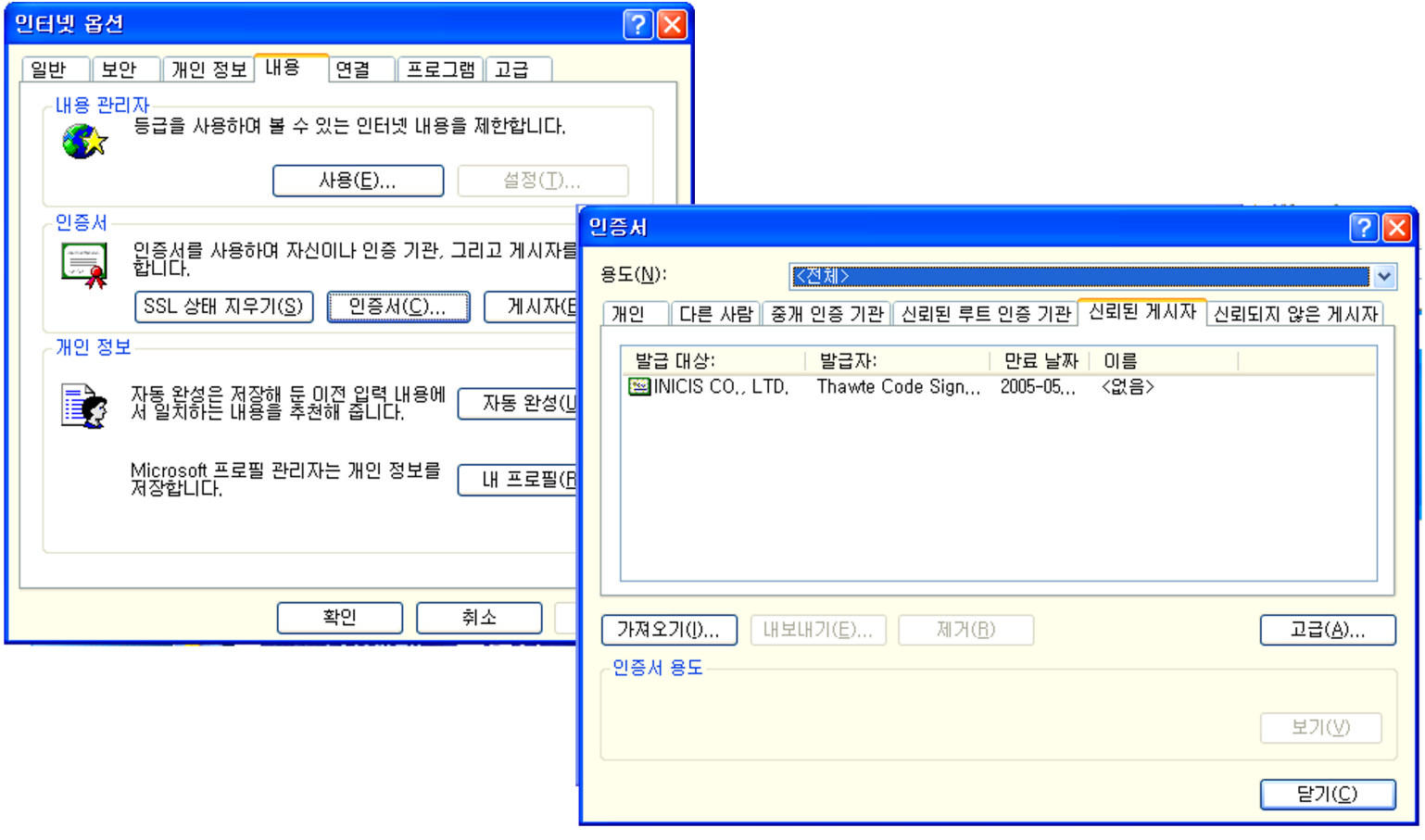
PK_{soft}

Program file | Sign(SK_{soft}, file)

Softforum Co. Ltd

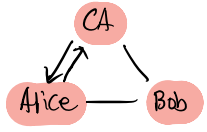
User

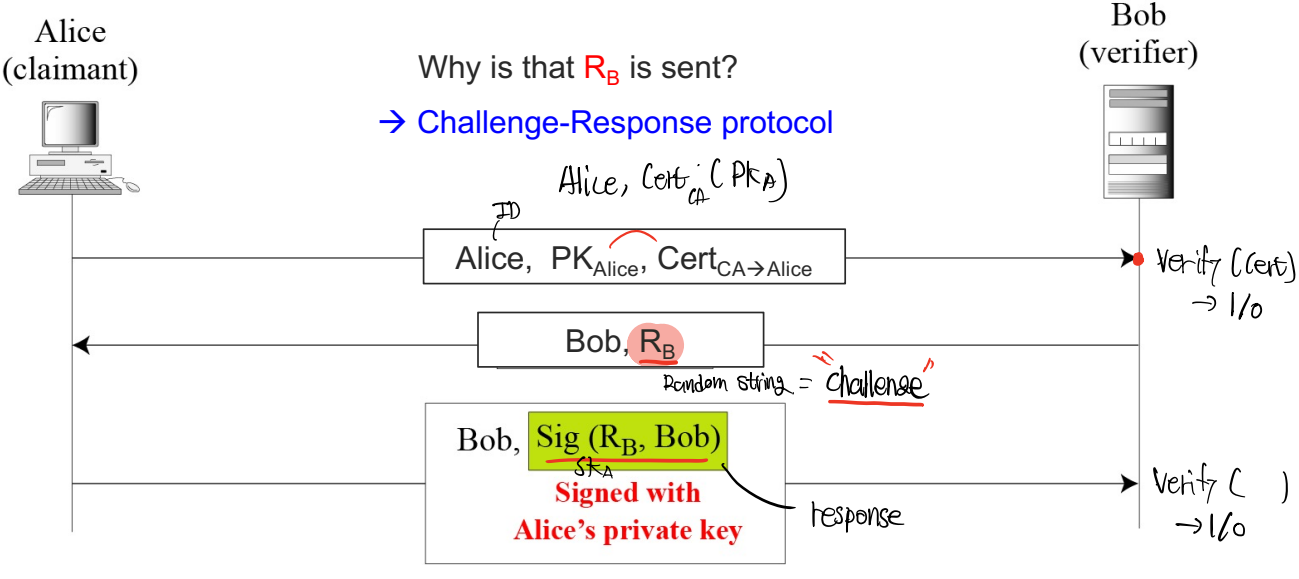
Integrity Check using DSS (3)



Authentication using DSS (1)

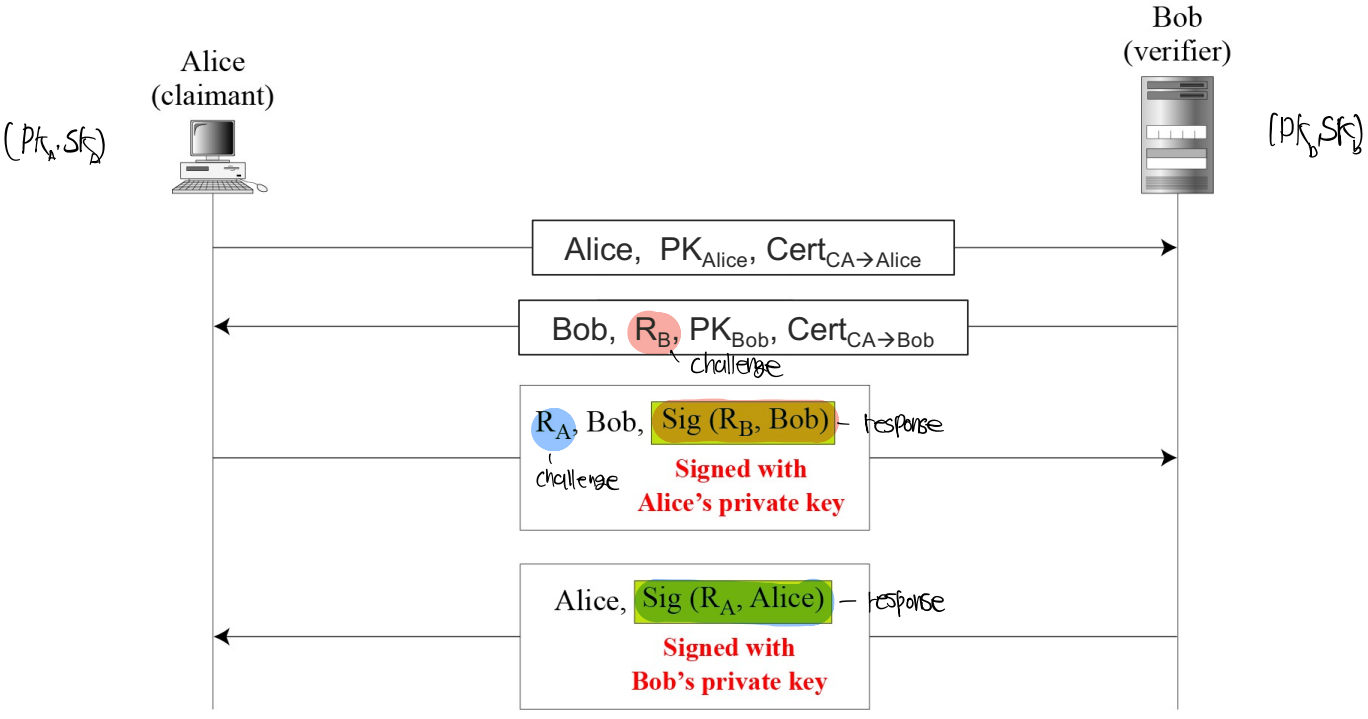
- Showing certificates can be used as "digital ID-card" (why?)
- Situation:
  - Alice has her own (PK, SK) and certificate related to PK
  - Alice is proving to Bob that "I am Alice"





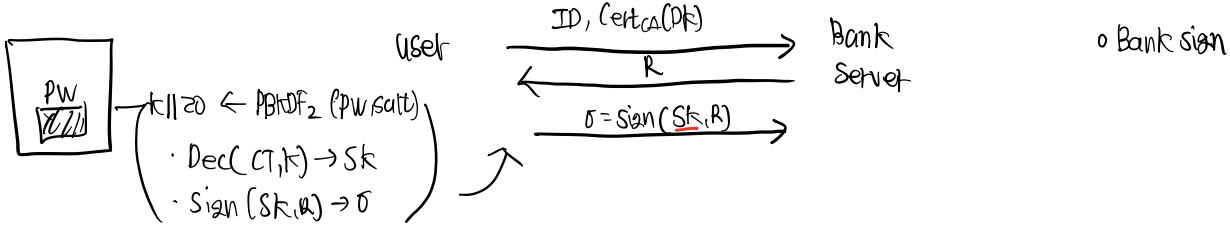
Authentication using DSS (2)

- Situation:
  - Alice and Bob have their own (PK, SK) and certificates
  - Both want to authenticate each other in bidirectional way



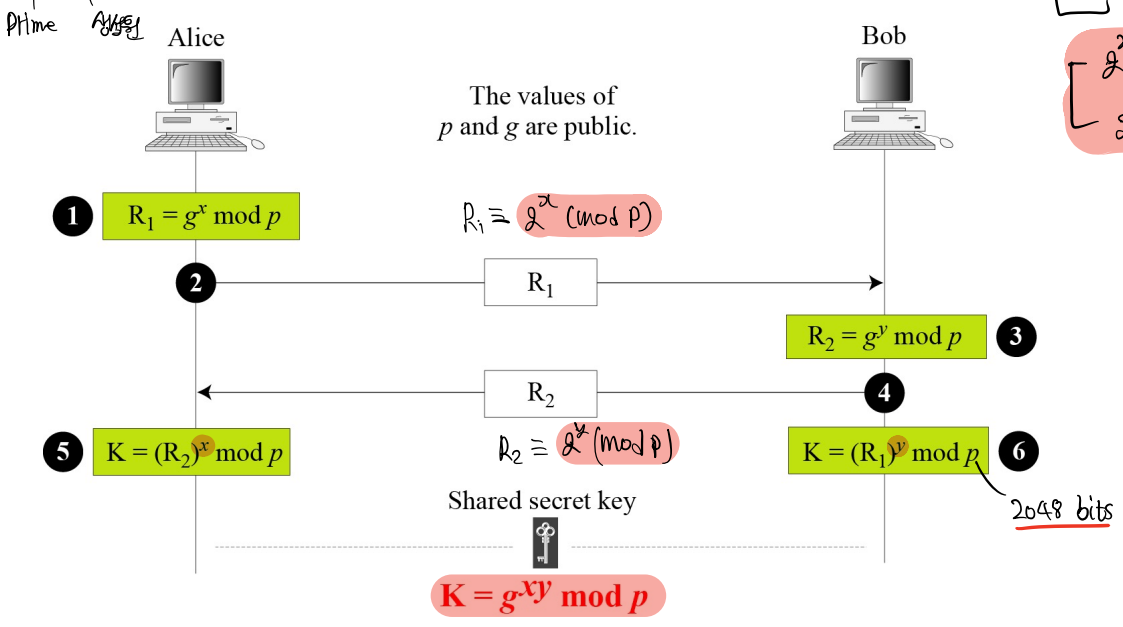
Authentication using DSS - example

- Entity authentication



Diffie-Hellman Key Exchange

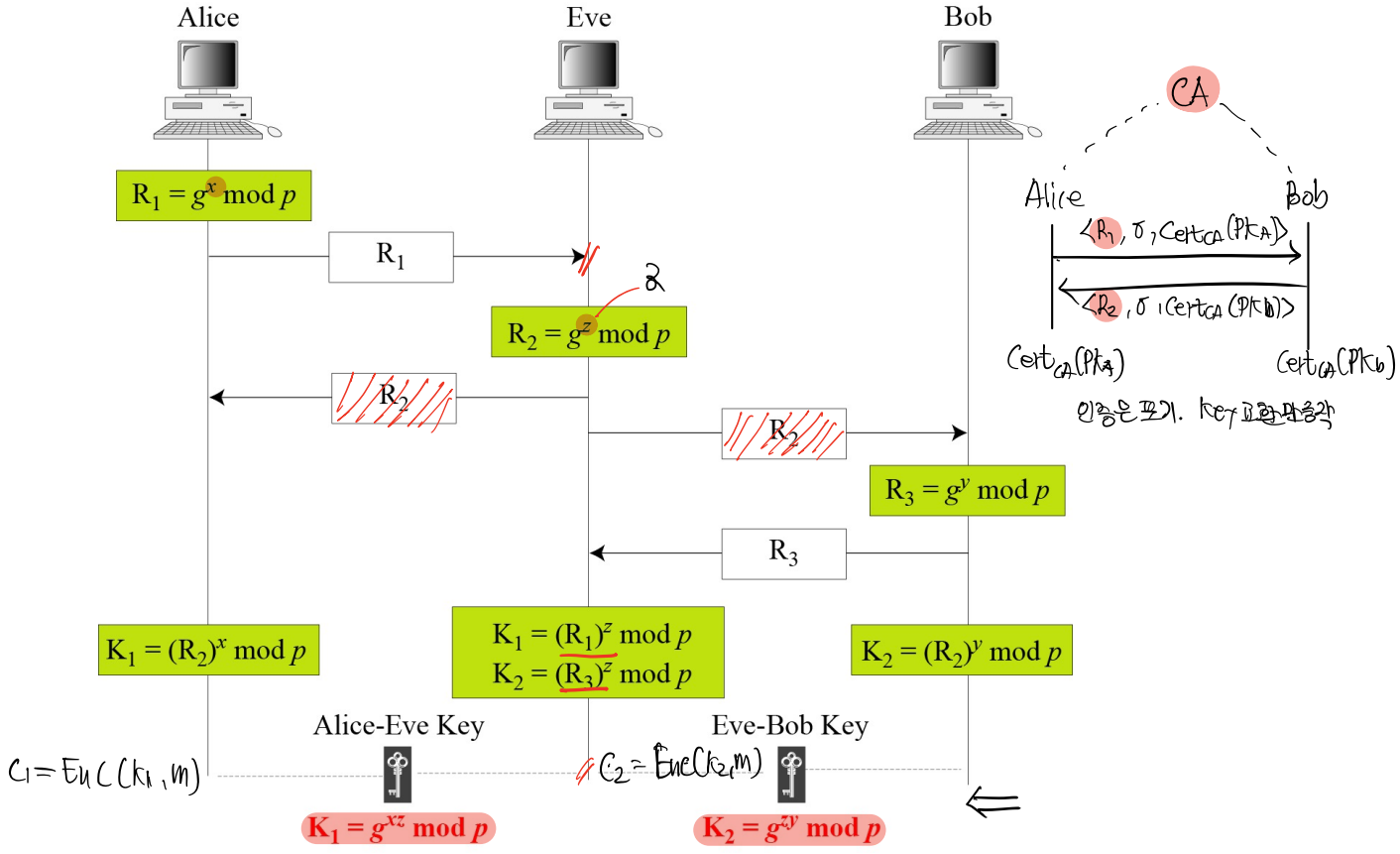
- How it works (by Diffie-Hellman, 1976)
  - (p, g) should be public in an authenticated manner



  - Even if a (passive) attacker can see g^x and g^y, it is infeasible to get $K=g^{xy}$
  - But, insecure against active attackers → man-in-the-middle attack

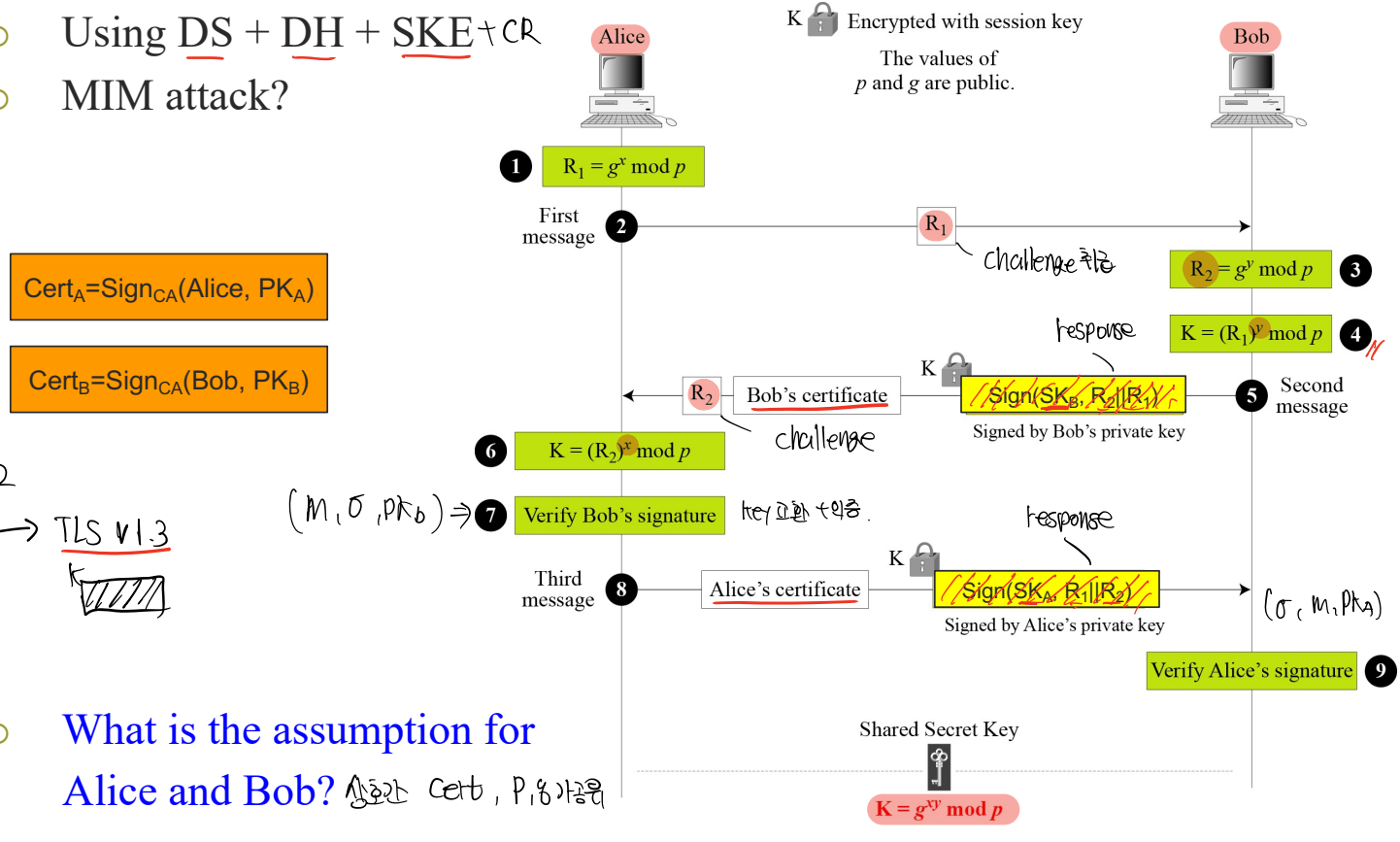
Man-In-the-Middle (MIM) Attack

- Active attack
  - Adversary Eve can intercept and modify messages being sent



How to prevent MIM attack (1)

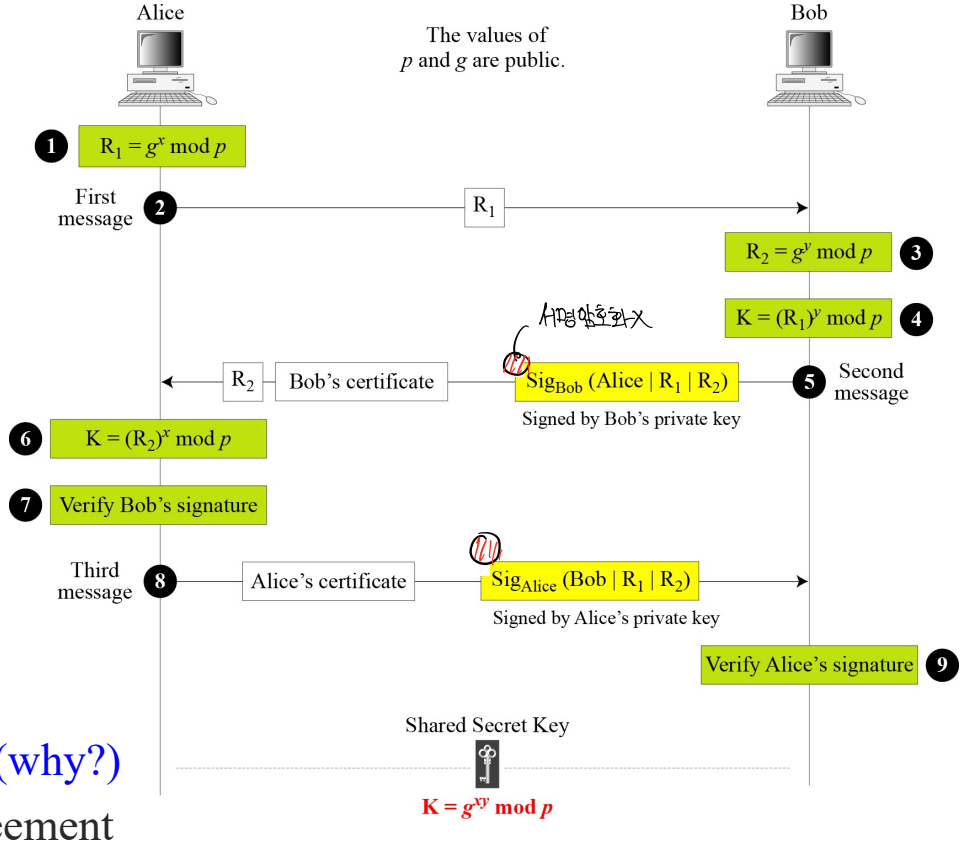
- Authenticated Diffie-Hellman key-exchange protocol
  - (a.k.a) Station-to-Station (STS) protocol 상호 인증 + key 교환
  - Using DS + DH + SKE + CR
  - MIM attack?



  - What is the assumption for Alice and Bob? 상호간 Cert, PK 사용

How to prevent MIM attack (2)

- Simplified STS protocol
  - What are differences?
  - Using DH + DS + CR
  - Mutual authentication (why?)
  - Authenticated key agreement



Authorization – ID card

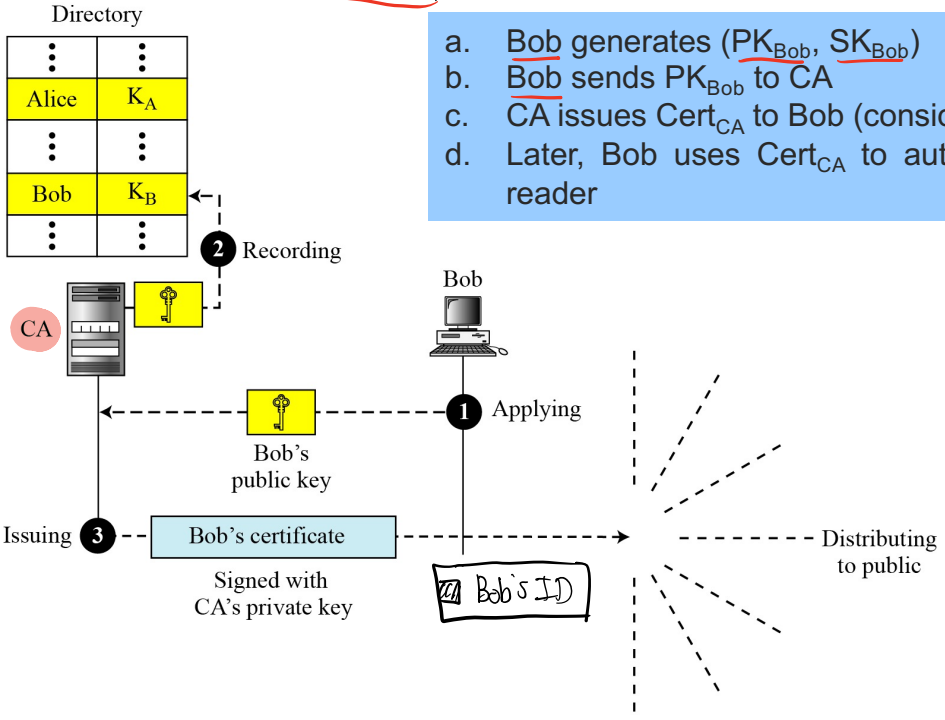
신분증또는 출입증.

- Certificate Authority (CA) in an organization
  - CA generates its own keys (PK_{CA}, SK_{CA})
  - $Cert_{CA}(PK_{Bob}) = PK_{Bob} || Sign(SK_{CA}, PK_{Bob} || Bob || access\ right)$

Treat it as a message signed

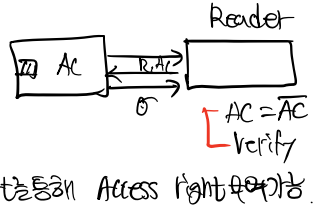
Bob || Access

권한

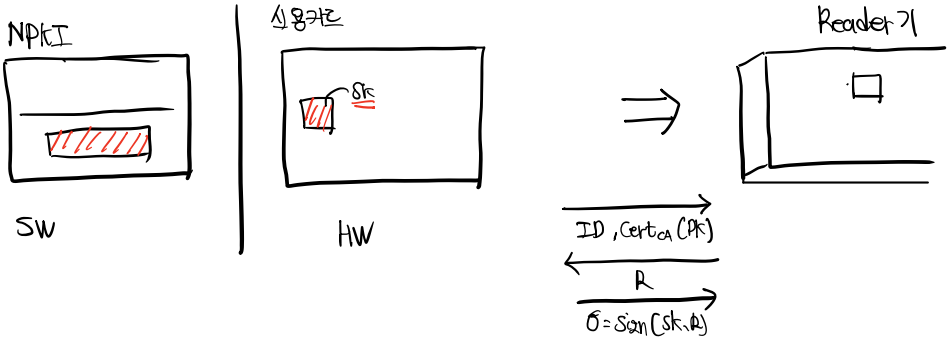


a. Bob generates (PK_{Bob}, SK_{Bob})
b. Bob sends PK_{Bob} to CA
c. CA issues $Cert_{CA}$ to Bob (considering access right)
d. Later, Bob uses $Cert_{CA}$ to authenticate himself to card reader

How should Card reader verify Bob's access right ?



Cert로통해 Access right확인가능.



Q & A